## CERTIFIED COPY OF PRIORITY DOCUMENT

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.
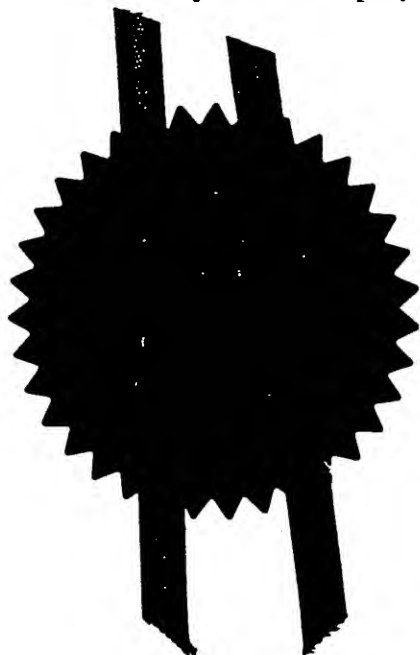
Signed

Dated   4 September 2001
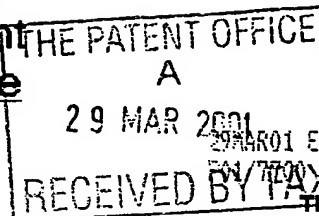
THIS PAGE BLANK (USPTO)

Patents Form 1/77

The
Patent
Office

1/77

THE PATENT OFFICE
A
2 9 MAR 2001
RECEIVED BY TAX

Patents Act 1977
Rule 16

29MAR01 E6 7950-1 D00611
00-0107882.3

The Patent Office
Concept House
Cardiff Road
Newport
South Wales NP10 8QQ

Request for grant of a patent

1.	Your reference	GB920010021GB1

2.	Patent application number *(The Patent Office will fill in this part)* 2 9 MAR 2001	**0107882.3**

3.	Full name, address and postcode of the or of each applicant *(underline all surnames)*	INTERNATIONAL BUSINESS MACHINES CORPORATION Armonk New York 10504 United States of America
	Patents ADP number *(if you know it)*	0051963 7001
	If the applicant is a corporate body, give the country/state of its incorporation	State of New York United States of America

4.	Title of the invention	PARSING MESSAGES WITH MULTIPLE DATA FORMATS

5.	Name of your agent *(if you have one)*	M J Jennings
	"Address for Service" in the United Kingdom to which all correspondance should be sent *(including the postcode)*	IBM United Kingdom Limited Intellectual Property Department Hursley Park Winchester Hampshire S021 2JN
	Patents ADP number *(if you know it)*	0778371500

		Country	Priority App No *(if you know it)*	Date of filing *(day/month/year)*
6.	If you are declaring priority from one or more earlier patent applications, give the country and the date of filing of the or of each of these earlier applications and *(if you know it)* the or each application number			

		No of earlier application		Date of filing *(day/month/year)*
7.	If this application is divided or otherwise derived from an earlier UK application, give the number and the filing date or the earlier application			

Patents Form 1/77

Patents Form 1/77

8.   Is a statement of inventorship and of right to grant of a patent required in support of this request? *(Answer 'Yes' if:*
   a) any applicant named in part 3 is not an inventor, or
   b) there is an inventor who is not named as an applicant, or
   c) any named applicant is a corporate body.)   Yes

9.   Enter the number of sheets for any of the following items you are filing with this form. Do not count copies of the same document

   Continuation sheets of this form

   Description   17

   Claim(s)   3

   Abstract   1

   Drawing(s)   3

10.   If you are also filing any of the following, state how many against each item.

    Priority documents

    Translations of priority documents

    Statement of inventorship and right to grant of a patent *(Patents Form 7/77)*   1

    Request for preliminary examination and search *(Patents Form 9/77)*

    Request for substantive examination *(Patents Form 10/77)*

    Any other documents *(please specify)*   Fee Sheet FS1

11.   I/We request the grant of a patent on the basis of this application
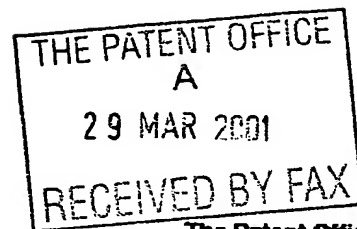
    Signature

    29 March 2001
    Date

12.   Name and daytime telephone number of person to contact in the United Kingdom   M J Jennings
    01962 818955

Patents Form 1/77

Patents Form 7/77

The Patent Office

7/77

Statement of inventorship and of right to grant of a patent

> THE PATENT OFFICE
> A
> 2 9 MAR 2001
> RECEIVED BY FAX

The Patent Office
Concept House
Cardiff Road
Newport
South Wales NP10 8QQ

1.	Your reference	GB920010021GB1

2.	Patent application number *(if you know it)*	2 9 MAR 2001 **0107882.3**

3.	Full name of the or of each applicant	INTERNATIONAL BUSINESS MACHINES CORPORATION

4.	Title of invention	PARSING MESSAGES WITH MULTIPLE DATA FORMATS

5.	State how the applicant(s) derived the right from the inventor(s) to be granted a patent	By employment and agreement

6.	How many, if any, additional Patents Forms 7/77 are attached to this form?	

7.

I/We believe that the person(s) named over the page (and on any extra copies of this form) is/are the inventor(s) of the invention which the above patent application relates to.

Signature   29 March 2001
          Date

8.	Name and daytime telephone number of person to contact in the United Kingdom	M J Jennings Tel: 01962 816725

Patents Form 7/77

THIS PAGE BLANK (USPTO)

Patents Form 7/77

Enter the full names, addresses and postcodes of the inventors in the boxes and underline the surnames

Michael <u>JOHNSON</u>
(UK resident)
c/o IBM United Kingdom Limited
Intellectual Property Law
Hursley Park
Winchester
Hampshire SO21 2JN
England

Patents ADP number *(if known)*

0811586 7001

Patents ADP number *(if known)*

If there are more than three inventors, please write their names and addresses on the back of another Patents Form 7/77 and attach it to this form

REMINDER

Have you signed the form?

Patents ADP number *(if known)*

Patents Form 7/77

GB920010021GB1   1

PARSING MESSAGES WITH MULTIPLE DATA FORMATS

FIELD OF INVENTION

5   The present invention relates to parsing of data, such as in a
messaging system in which messages may include a number of different data
formats.

BACKGROUND

10

         Parsing of data typically involves an initial lexical analysis step
which involves breaking up the input data into logically separate
components, such as field names, constants, operators (the lexical analyser
outputs a string of 'tokens') and then a syntactic analysis step which
15   involves processing the tokens to determine the syntactic structure of the
data as defined by a grammar. A lexical analyser may also remove redundant
spaces and may deal with character-set mappings (e.g. replacing upper case
letters with lower case). The term parser is used to refer to a program
which performs such analysis. The output of a syntax analyser may be a
20   syntax tree (also known as a parse tree) which represents the syntactic
structure of the string of data being parsed. Parsing is well known in the
context of compilers, but is a required step for many data processing
operations. For example, a message processing program may need to parse an
input message to model the message structure before it can begin processing
25   the message – for example to understand the structure and format of the
message before performing format conversion. This may include separating a
message descriptor into a set of fields comprising name-value pairs so that
the different values in the named fields can be processed, and similarly
separating a stream of bits comprising the message data into name-value
30   pairs so that the data can be processed.

         It is now very common for a computing network to integrate many
heterogeneous systems, and individual messages sent across these networks
may include different data formats within headers inserted by the different
35   systems through which the message passes. It is therefore important for a
network-wide messaging service to be capable of handling a number of
different data formats within a single message, to support the increasing
requirement for business and system integration. It is a feature of some
existing messaging products to be able to parse an incoming message which
40   includes a number of different format components, splitting the message
into its differently formatted constituent parts and separately parsing
these parts to generate an output message for further processing.

   In the past, these messaging products have relied on predefined
message formats and either a generic parser or a single parser-selection
process which has access to a repository of message formats. Either the
generic parser analyses all components itself or a process scans through
5   the message to identify the differently formatted components, comparing the
identified formats with those in the repository, and then a selector
assigns each component to a specific parser which is capable of performing
syntactic analysis for that component.

10   This approach has proven satisfactory for situations in which only a
small number of formats are possible and the format and sequence of all the
message components is known in advance, since this knowledge enables the
selection of appropriate parsers for the different components. It may also
be possible in some cases, although inefficient, to rely on a generic
15   parser to perform an initial analysis and resulting parser-selection before
the main syntactic analysis begins. However, this would require the generic
parser to be capable of analysing all data formats within each message so
that the first scan of the message could break the message into components
and could provide the information to determine which specific parser should
20   handle which components. It may be very difficult to implement such
capability within a single generic parser. Secondly, separating syntactic
analysis into separate first and second steps would entail processing
delays and would tend to duplicate some of the analysis.

25   An alternative to this initial step of a generic parser analysing the
whole message is for individual selected parsers to perform syntactic
analysis of specific parts of a message, including identifying the data
format or data type of the next message component in the sequence of bytes
which makes up the message. In this case, a parser-selection process can
30   identify the format of a first component and select a first specific
parser; the selected parser can then parse this first component, identify
the format or type of the next component and send this format or type
information to the parser-selector to invoke another specific parser for
this next message component. As noted above, such solutions rely on
35   knowledge of predefined message formats and predictable sequences of
message components. This has proven satisfactory for handling differently
formatted message headers, if the selected parsers are given the knowledge
of which field to read to determine the format of the next message
component, and if the data within the body of the message has a single
40   format such that a single parser can parse the entire contents of the
message body. The class name of the required parser can be included in a
format field of a message descriptor or another header of the message, and
this can be read to select and then invoke the appropriate parser.

A problem arises with the above solution when a message body includes multiple nested data formats, since then the reliance on the single parser-selector to call the appropriate program at the required time involves an excessive number of·communication flows between the specific selected parsers. and the parser-selector. It also requires the parser-selector to be able to invoke a suitable specific parser for all formats and for unpredictable nested structures, such that the selector needs a detailed knowledge of an ever increasing number of formats. The problems of this approach will become clear in future as the number of message data formats and the complexity of message contents increases with increasing systems and business integration.

Thus, there remains a requirement for an efficient solution to parsing of messages which include multiple data formats, especially when the different formats can be nested within one another and the structure of the message is not known in advance of its receipt and analysis by a messaging program. The problems of known solutions are especially acute for messages in which either individual components or the sequence of components do not have a predictable structure, since then a message analysis is required as part of the run-time operation of the messaging program before it is possible to select a parser to analyse a next component of the message.

SUMMARY OF INVENTION

In a first aspect, the present invention provides a data processing apparatus for processing messages which may contain a plurality of data formats, the apparatus including a set of selectable parsers, each adapted for analysing a specific set of one or more data formats, wherein a plurality of said set of parsers each include: means for parsing a first component of a message, means for identifying the data format of a second component of the message, and means, responsive to said identification, for selecting another one of said set of parsers and for invoking the selected parser to parse the second message component.

This preferably includes the capability of a first selected parser, to which a portion of a message has been allocated for parsing, to identify subcomponents within this message portion which should be allocated to a different parser and to select and invoke a parser using subcomponent format information within the message. The first selected parser preferably passes only a specific subcomponent to the different parser and keeps control for analysing format information for other subcomponents to select and invoke an appropriate parser for those subcomponents.

This invention changes the distribution of responsibilities compared with existing parsing solutions, producing an hierarchical tree of parsers to handle nested-format messages instead of the horizontal chain of parsers controlled by a single selector process (which is the result of known

5   solutions). The depth of the hierarchy can be determined in response to the structure of a received message rather than being hard coded into the parser program. The invention mitigates the inefficiencies of solutions which rely on a single generic parser to handle all of the analysis of message components or to perform a partial analysis to determine which

10   specific parsers to use. It also mitigates the inefficiencies of solutions in which a plurality of parsers are always required to call a generic parser-selector to invoke another specific parser.

Since an analysis step performed by a generic parser cannot be

15   optimised for all data formats, messaging systems which rely on generic parsers will be unable to efficiently handle multiple-format messages in a heterogeneous network. As systems integration requirements and the number of different message formats increase, the inadequacies of such systems will become clear. Similarly, reliance on a single parser-selector process

20   will prove inadequate as the number of message data formats and the possibility of nested-formats and unpredictable structures within the body of a message increases.

Hence there are significant benefits in the solution of the present

25   invention which enables specific parsers to perform the selection and instantiation of a next parser without having to make a call to a generic parser or parser-selector to perform this instantiation. The invention enables a reduction in the cost and potential delays of inter-process communication flows and avoids having to continually update the generic

30   parser or parser-selector to provide capabilities to cope with new formats and structures. Nevertheless, the solution of the present invention is in contrast to the current technological trend of relying on a single generic parser or a single specialised parser selection process. The invention requires a recognition that the limitations of current systems will result

35   in significant problems when parsing complex nested-format messages, and a recognition that changing the structure of and distribution of responsibilities within a parsing mechanism by adding parser-selection and parser-invocation capabilities to specific selectable parsers will provide significant benefits which outweigh the costs.

40

A specific example of the problems of known prior art solutions arises when a message to be parsed contains a component in SAP AG's IDoc format (IDoc is the short-form name of the 'Intermediate Documents' exchanged within SAP's R/3 system, and the name is also used to describe

GB920010021GB1   5

the data format of these documents. SAP and R/3 are trademarks of SAP AG).
An IDoc component contains a control structure component (DC) and a
variable number of data components (DD) and it is only possible to
determine the number of DDs during parsing. There may also be multiple
5   IDocs within a single message stream. A typical generic parser would not be
capable of separating an IDoc into its constituent parts and analysing
those parts. Similarly, it would be inefficient to rely on a single generic
parser-selection process to assign the DC and DDs to different parsers. A
preferred embodiment of the present invention allows a specific IDOC parser
10   to be selected in response to a parser of a message descriptor or header
identifying that the next component has the IDoc format. The IDOC parser
can then analyse the DC component and select and instantiate a further
parser for handling each of the DD components. After each DD component is
handed to a newly instantiated parser, the IDOC parser retains
15   responsibility for identifying the next component (whether a DD or another
DC) and selecting the appropriate parser. This ability to instantiate a
parser to handle a subcomponent of an allocated portion of a message is a
significant difference from control always being returned to a single
parser-selector for each component.
20

As well as enabling incoming IDoc data to be represented to a message
processing program in a format that the program can understand and
manipulate, the IDOC parser according to a preferred embodiment of the
present invention also supports creation of output messages in the SAP IDoc
25   format. This two-way format conversion capability supports integration
between SAP systems and a message processing program which includes the
IDOC parser.

To implement IDoc parsing capability within a single generic
30   parser-selector, and to add similarly specific functionality to the single
selector for other data formats such as SWIFT messages, would result in
excessive program complexity and require repeated re-coding of the selector
to support new message types. The SWIFT message format, defined by the
Society for Worldwide Interbank Financial Telecommunication, is another
35   example of a message format where the sequence of components is not fully
known in advance.

As the requirements for systems and business integration extend to
cover additional systems, and as new data formats are created, the present
40   invention provides an extensible solution to which new parsers can be added
which have integrated parser-selection capabilities. This is easier to
update and maintain than known solutions in which the new capabilities have
to be added to a generic parser or a single parser-selector.

GB920010021GB1   6

In a second aspect, the invention provides a method of parsing a
message containing a plurality of data formats, the method comprising:
identifying the data format of a first component of the message; responsive
to said identification, selecting and invoking a first parser to parse the
5   first component; and using said first parser to identify the data format of
a second component of the message and, responsive to said identification,
using said first parser to select and invoke a second parser for parsing
the second message component.

10   In a preferred method, the second message component can be a
subcomponent of the first component, and the first selected parser is able
to assign a specific chunk of a message bit stream (corresponding to this
second subcomponent) to the second parser while retaining control for
selecting and invoking a parser for subsequent message components. In the
15   example of IDoc message components mentioned above, the IDOC parser assigns
a DD to a specific parser while retaining responsibility for identifying a
next message component (whether a DC or a specific format DD) and for
selecting and invoking a next parser (for a DD) or parsing the next
component itself (in the case of a DC).

20
In a third aspect, the invention provides a message processing system
including a parser selector and a set of selectable parsers, each
selectable parser being adapted for analysing a respective set of message
data formats, wherein said set of selectable parsers are each selectable in
25   response to identifying a message data format within the respective set,
and wherein at least one of said selectable parsers includes: means for
parsing a first component of a message having a message data format within
the respective set; means for identifying the data format of a second
component of the message; and means, responsive to said identification, for
30   selecting another one of said set of parsers and for invoking the selected
parser to parse the second message component.

The above-described selectable parser preferably comprises an IDOC
parser for parsing message data in the IDoc format, including means for
35   parsing a DC component of an IDoc, means for identifying a DD component,
and means for selecting and invoking another parser for parsing the
identified DD component.

In a fourth aspect, the invention provides a computer program for
40   controlling the operation of a data processing apparatus on which it runs
to perform a method as described above. The computer program may be made
available as a program product comprising program code recorded on a
computer readable recording medium.

GB920010021GB1   7

BRIEF DESCRIPTION OF DRAWINGS

Preferred embodiments of the invention will now be described in more detail, by way of example, with reference to the accompanying drawings in
5   which:

Figure 1 shows a message broker within a messaging network, including a parsing solution according to an embodiment of the present invention;

10   Figure 2 shows a simple example message structure;

Figure 3 is a schematic representation of message flows between a SAP R/3 system and a message broker such as IBM's MQSeries Integrator broker;

15   Figure 4 is a representation of the hierarchical parser's view of a message which includes an IDoc; and

Figure 5 is a representation of a sequence of steps according to an embodiment of the invention.

20

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

Figure 1 is a schematic representation of a messaging network in which a number of application programs 10 are communicating via messaging
25   manager programs 20 which provide message delivery services and via a message broker 30 which implements a set of rules to provide message routing and message transformation (and message creation in response to certain received messages). The message broker 30 and messaging manager programs 30 enable diverse applications to exchange information in unlike
30   forms - the brokers handling the processing required for the information to arrive in the desired place in the correct format and the messaging manager programs handling the complexities of network communications to provide assured delivery of messages across the network between different computer systems. Such messaging services are provided by the commercially available
35   MQSeries family of products from International Business Machines Corporation (MQSeries is a trademark of IBM Corporation).

IBM's MQSeries Integrator products support business and application integration, using message brokers 30 to manage the flow of information
40   between applications. The brokers are a resource which hosts and controls message flows 40 which implement business processes. Applications send messages to a message flow, which is a sequence of message processing nodes that each implement rules (a set of processing actions), to process the messages and generate one or more outputs for sending to target

applications using MQSeries messaging and queuing communications. IBM's
MQSeries Integrator products include a Control Centre which provides a user
interface for creating message flows by selecting and connecting message
processing nodes. IBM's MQSeries Integrator products are described in more
5   detail, for example, in MQSeries Integrator v2.0.1 Introduction and
Planning, IBM Document GC34-5599-01.

Among many other capabilities, the brokers provide the capabilities
to:
10

- Route a message to several destinations using rules that act on the
  contents of one or more fields in a message or message header; and
- Transform a message so that applications using different formats can
  exchange messages in their own formats. The broker knows the
15   requirements of each application (for example, whether personal names
  should have the surname first or last, with or without middle initials,
  upper or lower case) and so it can transform the message to the correct
  format (changing the order of fields, the byte order, the language, and
  so on).
20

Within a message flow 40, the action to be taken can be defined
according to the message structure, the message topic, or the data within
the message. Alternatively, the originator of the message or its
destination may be significant. Any combination of one or more of these
25   attributes can define the rules by which messages are processed. A message
flow can process one message in several ways to deliver a number of output
messages, perhaps with different format and content, to a number of target
applications.

30   Each message flowing through a message broker has a specific
structure, which is important and meaningful to the applications that send
or receive that message. Message structure information as used in IBM's
MQSeries Integrator products comprises a message domain, message set,
message type, and wire format of the message. Together these values
35   identify the structure and format of the data the message contains. Every
message flow that processes a message conforming to this structure and
format must understand it to enable the message bit stream to be
interpreted.

40   The message structures which can be handled by the message brokers
comprise those which are predefined within one or more message repositories
60 as well as self-defining message structures (such as message data in XML
which is a string of name-value pairs such the parser can simply locate a
specified name in a message and extract the data value). The message

GB920010021GB1   9

repositories hold information for a number of message formats defining the
arrangement of named data fields in messages of the respective format. Then
a format name lookup step can extract the message format template which can
be applied to any message having that format to generate a view of the
5   message as a sequence of name-value pairs which can be interpreted and
manipulated.

When a message is to be processed by a message flow, it is first
necessary to decode the message bit stream using one or more message
10   parsers 50 to enable the various components of the message to be
understood. Message type and format information for messages predefined in
the message repositories is typically included in the messages' headers,
and so a parser can recognise the message structure and format when the
message is received. Messaging products such as IBM's MQSeries Integrator
15   product are known to include a number of message parsers 50 which can parse
known message structures including specific headers. However, with
increasing systems and network integration, it is a requirement for these
solutions to be extensible to add new parsers for new message structures
and new data formats within a message. A particular extensible solution
20   which is implementable within computer program products such as IBM's
MQSeries Integrator products, and which satisfies this message broker
parsing requirement will now be described in detail.

Figure 2A shows an example message structure of a message as taken
25   from an input message queue. This comprises a message descriptor 100 (MD),
one or more additional message headers 110 (in this case RFH2), and a
message body 120. Normally the message data within the body part of the
message has a single format, and so a single parser will handle the parsing
of the entire data contents. Nevertheless, as the bit stream of the message
30   is analysed to identify the separate MD, headers and body components, each
of these components will be assigned to an appropriate parser.
Additionally, the message as modelled by parsing will include a Properties
component which is generated from data provided by the input node of the
message flow (see next paragraph - this is possible if only messages having
35   certain properties are input to this input node) and/or from data extracted
from the message headers. The Properties component will be handled by an
appropriate parser which is typically different from the parsers handling
the MD, headers, and body. If the message body contains multiple nested
formats, then multiple parsers will also be required to parse the body. The
40   implementation of parser selection and invocation for handling nested
message formats will be described below.

Within a simple example message flow of a message broker comprising
an input node, a processing node and an output node, the processing node

may be a Compute node which transforms a message from one format to another, so that sending and receiving applications can communicate with each other using their own formats. Such a simple message flow is shown in Figure 3. The input node requires the message to be parsed to understand
5 its structure before the Compute node can perform its format transformation.

On receipt of a message, the message broker 30 must pass the message bit stream (an array of bytes which make up the message) to a message
10 parser 50. Consider the previous example of an incoming message which comprises:

* A message descriptor (MD)
* An RFH2 header
15 * An XML data portion

A parser selector 80 of the broker 30 sends the MD component to an MD parser 50, which accesses a structure template for MDs which is stored in the message repository 60 and applies this to the received MD to model it
20 as a sequence of ordered name-value pairs. The MD parser then reads one or more predefined message fields identifying the next message component's type and/or its format, compares this information with a list of component types/formats and selects and invokes another parser 50 which has a predefined responsibility for handling parsing of components having this
25 type and/or format. The MD parser also specifies what portion of the bit stream it has consumed to indicate where a next selected parser 50 should begin. If the next component is the RFH2 header, then this component is given to a specific RFH2 parser. The RFH2 parser applies a stored RFH2 template to parse the header, modelling it as a sequence of name-value
30 pairs, and then this RFH2 parser reads one or more fields relating to the next component. If the next component is the XML data portion, then the RFH2 parser invokes an XML parser in response to identifying that the component comprises XML. Since the templates stored for the MD and message headers determine what is the last field within their respective message
35 component, the MD and RFH2 parsers can easily determine when to invoke the next parser.

Once the relevant parsers within the set of available parsers have been invoked to handle their respective chunks of the message bit stream
40 and to create a syntax element tree, this syntax element tree can then be manipulated by the message broker's processing nodes. For example, the formatting 'Compute' node constructs an output message the elements of which can be defined using an SQL expression and are based on elements of the input message. Other processing nodes' output message elements may be

based on elements of the input message and data extracted from an external database.

5   The above description shows the sequence of operations of an embodiment of the present invention in which a selected parser selects and invokes a next parser for handling the next component of the message, but the above description relates to an example with a single format message body. Many received messages will not be this simple.

10   Referring to Figure 2B, if a message's body component 120 comprises an IDoc, then the MD 100 or another message header 110 will include an identification of this component type (for example showing 'IDOC' in the format field of the MD). This information is used during parsing of the message to invoke a dedicated IDOC parser 50 and to indicate which format
15   dictionary 60 is to be used. Thus far, there is no conceptual difference from the XML body example in which selected parsers can select and invoke further parsers. However, markup languages such as XML are self-defining and so a single XML parser can be used. An IDoc, which is not self-defining in the markup sense, also does not have a predicable number of data
20   components. In particular, a control structure DC component 130 may be followed by one or many data segment DD components 140. The last one of this sequence of DDs may be the end of the message, or it may be followed by another DC and one or more DDs, or there may be additional non-IDoc message components. Therefore, an analysis of the IDoc structure is
25   necessary to determine which parser to use for which sub-component. In this case, it is desirable for the IDOC parser to control the selection and instantiation of particular parsers to handle subcomponents of the bit stream for which it is responsible - the IDOC parser building an hierarchical tree of parsers.
30

In brief, the Version 4 IDoc structure comprises fixed size structures, the first of which is the DC control structure which is 524 bytes long. The one or more subsequent segment data DD's are each 1063 bytes long. Each DD structure is composed of name-value pairs plus a 1000
35   byte final field which holds the segment data. An IDoc can hold many different message types and each message type can have many different segments.

The IDOC parser is invoked in response to identification of the IDOC
40   type information in a message descriptor or message header field. Firstly, the IDOC parser parses the DC component, applying its knowledge of IDoc structures (a DC template) to generate a sequence of name-value pairs. Next, the IDOC parser investigates the first field of the DD which follows the DC. This provides an identification of a specific message format within

the dictionary of message formats which is associated with the IDOC parser. This message format identifier and dictionary name is then passed to a Message Repository Manager which both implements the respective parser and holds the format templates which are required by the parser programs. This

5   Message Repository Manager comprises a collection of parsers and a dictionary of formats in which individual format names are associated with predefined format templates. The Message Repository Manager is thus instantiated to run the relevant parser for the 1063 bytes of the current DD, to model this data for subsequent processing. The IDOC parser then

10   investigates the first field of the next component of the message, which may be a DC or DD, and then instantiates the required parser to handle this next component. If it is a requirement for the message broker to be able to handle messages which include additional data subsequent to the last DD of the IDoc, then the IDOC parser will also be required to investigate the

15   format of the next component to invoke an additional parser.

Thus, the IDOC parser takes input messages in valid IDoc format and creates logical message tree structures which can be interpreted by the broker's message flows. Similarly, this IDOC parser can take a logical

20   message tree created by a message flow and produce the data portion of a message broker's message in IDoc format. That is, once a Compute node, for example, has finished manipulating the syntax element tree which represents the message, the IDOC parser will be called once again to rebuild the bit stream at the output node of the message broker's message flow.

25

In a particular implementation of the invention for parsing logical message trees within a message broker, all such trees have a basic structure which has a single high-level element known as the root element. The user data found in the body of the message has a high-level element

30   which is one of the child elements (typically the last child element) of the root element as well as additional elements at a lower level of the hierarchy which correspond to the sub-components of the IDoc. For a SAP IDoc message, the name of the high level element is 'IDOC' to match the message domain supported by the parser, since the IDOC parser registers

35   with the broker for a message domain of 'IDOC'. The IDOC parser then attempts to parse any input message that is read by the message broker and has this message domain specified either in the RFH2 header or another message header or in the input node of the particular message broker's message flow. The parser will thus create a logical message tree that

40   reflects the contents of the message, with the top-level element of the body having the name 'IDOC' and the rest of the data for this IDoc structure built as a logical data structure unde this high-level element.

GB920010021GB1   13

        Figure 3 provides a schematic representation of a SAP R/3 system
communicating with a message broker such as IBM's MQSeries Integrator
product via a communication link and a link program component (such as the
MQSeries link for SAP R/3 program) which handles placing of a message
5   received from the SAP system onto a queue under transactional control (i.e.
Using a transaction ID for a received message to ensure one full copy only
of each message is placed on the queue). The input node of the message
broker's message flow has been configured to read from this queue. The
default message domain of the input node is set to 'IDOC'. A syntax element
10   tree is generated by the parsing processes as described above. An message
including an IDoc sent from a SAP R/3 system via the link program component
and then parsed as described above would be represented by the parsing
processes including the hierarchical IDOC parser as shown in figure 4. All
messages from the link program component connecting the SAP R/3 system to
15   the message broker include an additional header not described above. This
SAPH header is assigned to its own parser. As stated previously, any number
of additional message headers and nested formats can be handled as
described above. The link program component also ensures that atomic
transactional principles are followed when sending messages from the
20   message broker to the SAP system, rolling back a message onto the broker's
output queue to enable a retry if it is not possible to deliver it on a
first attempt.

        A list of field names of DC and DD structures which are understood by
25   the IDoc parser of a first implementation of the present invention are
shown below. These are documented in the form in which they would be used
in a SET statement of eSQL. For example:

    SET OutputRoot.Properties = InputRoot.Properties;
30   SET OutputRoot.MQMD = InputRoot.MQMD.

Control section (DC) fields

All fields must be specified and set
The syntax is <rootname>.<Parser Name>.<folder name>.<field name> =
35

```
SET "OutputRoot"."IDOC"."DC"."tabnam" =
SET "OutputRoot"."IDOC"."DC"."mandt" =
SET "OutputRoot"."IDOC"."DC"."docnum" = '0000000000000001';
SET "OutputRoot"."IDOC"."DC"."docrel" = '45B';
SET "OutputRoot"."IDOC"."DC"."status" = '30';
SET "OutputRoot"."IDOC"."DC"."direct" = '1';
SET "OutputRoot"."IDOC"."DC"."outmod" = '4';
SET "OutputRoot"."IDOC"."DC"."exprss" = '';
SET "OutputRoot"."IDOC"."DC"."test" = '';
```

GB920010021GB1   14

```
SET "OutputRoot"."IDOC"."DC"."idoctyp" = 'MATMAS01';
SET "OutputRoot"."IDOC"."DC"."cimtyp" = '';
SET "OutputRoot"."IDOC"."DC"."mestyp" = '';
SET "OutputRoot"."IDOC"."DC"."mescod" = '';
SET "OutputRoot"."IDOC"."DC"."mesfct" = '';
SET "OutputRoot"."IDOC"."DC"."std" = '';
SET "OutputRoot"."IDOC"."DC"."stdvrs" = '';
SET "OutputRoot"."IDOC"."DC"."stdmes" = '';
SET "OutputRoot"."IDOC"."DC"."sndpor" = '';
SET "OutputRoot"."IDOC"."DC"."sndprt" = '';
SET "OutputRoot"."IDOC"."DC"."sndpfc" = '';
SET "OutputRoot"."IDOC"."DC"."sndprn" = '';
SET "OutputRoot"."IDOC"."DC"."sndsad" = '';
SET "OutputRoot"."IDOC"."DC"."sndlad" = '';
SET "OutputRoot"."IDOC"."DC"."rcvpor" = '';
SET "OutputRoot"."IDOC"."DC"."rcvprt" = '';
SET "OutputRoot"."IDOC"."DC"."rcvpfc" = '';
SET "OutputRoot"."IDOC"."DC"."rcvprn" = '';
SET "OutputRoot"."IDOC"."DC"."rcvsad" = '';
SET "OutputRoot"."IDOC"."DC"."rcvlad" = '';
SET "OutputRoot"."IDOC"."DC"."credat" = '';
SET "OutputRoot"."IDOC"."DC"."cretim" = '';
SET "OutputRoot"."IDOC"."DC"."refint" = '';
SET "OutputRoot"."IDOC"."DC"."refgrp" = '';
SET "OutputRoot"."IDOC"."DC"."refmes" = '';
SET "OutputRoot"."IDOC"."DC"."arckey" = '';
SET "OutputRoot"."IDOC"."DC"."serial" = '';
```

Data section (DD) fields

To access each DD segment use the array suffix ie. DD[1] , DD[2] etc.
Note the use of the '2' sufix to give unique field names on the mandt and
docnum fields

```
SET OutputRoot.IDOC.DD[I].segnam  = 'E2MAKTM001';
SET OutputRoot.IDOC.DD[I].mandt2  = '111';
SET OutputRoot.IDOC.DD[I].docnum2 = '99999999999999111';
SET OutputRoot.IDOC.DD[I].segnum  = '111000';
SET OutputRoot.IDOC.DD[I].psgnum  = '000111';
SET OutputRoot.IDOC.DD[I].hlevel  = '11';
```

GB920010021GB1   15

Segment fields

Use the 'sdatatag' keyword to indicate to the parser that it is the
element that contains the segment data which is to be manipulated.   The
5   MRM indicates that the MRM will handle the transformation.   The final field
identifies the actual field of the segment.
The final line is the filler for the segment as an incoming IDoc to SAP
must have each segment 1000 bytes long.

```
10    SET OutputRoot.IDOC.DD[I].sdatatag.MRM.msgfn = '006';
      SET OutputRoot.IDOC.DD[I].sdatatag.MRM.spras = 'E';
      SET OutputRoot.IDOC.DD[I].sdatatag.MRM.maktx = 'Buzzing all night';
      SET OutputRoot.IDOC.DD[I].sdatatag.MRM.msgfn = '006';
      SET OutputRoot.IDOC.DD[I].sdatatag.MRM.spras_iso = 'EN';
15    SET OutputRoot.IDOC.DD[I].sdatatag.MRM.fill954 = '    ';
```

Note that the traditional way of parsing a bit stream is to traverse
the incoming bit stream from left to right, paring the respective sections
of the bit stream to the appropriate parser until all the bit stream is
parsed.   This in effect is a horizontal pass of the data.   According to the
20   embodiment of the present invention described above, an hierarchical tree
of parsers may be created since one or more of the selected parsers is able
to call other parsers to process subcomponents of the portion of the
message bit stream it is handling, or to call a parser to handle the next
component of the message subsequent to that portion. Thus, an incoming bit
25   stream is passed to an appropriate parser and this parser then passes
sections of this bit stream to other parsers before returning back to the
original bit stream to have the remains of the unclaimed bit stream passed
to the next appropriate parser.   This hierarchical parsing approach allows
a message broker to model data which is not known until runtime, without
30   excessive communication flows to and from a single selector process and
without the overhead of having to update a generic parser or generic
selector each time a new data format is added. This approach enables a
dedicated parser to be given detailed knowledge of the structures and
formats of a particular type of message for which it will be responsible.
35   Implementing this level of detailed knowledge of many different message
types in a single parser or parser-selector would be a very inefficient.

While an embodiment of the present invention has been described above
using the example of IDocs, aspects of the invention are equally applicable
40   to other message data formats such as messages which include data in the
SWIFT message format where the sequence of components is not fixed or other
message structures and formats which are not fully predefined.

GB920010021GB1   16

The invention is also applicable to data processing environments other than message brokers in an heterogeneous network, although the invention is especially advantageous for that environment. Thus, one aspect of the present invention provides a parsing program comprising a set of
5   selectable parsers which are each adapted for parsing a specific set of one or more data formats, wherein a plurality of the set of parsers have the following capabilities:

• Parsing a first component of a bit stream;
10   • Identifying the data format of a second component of the bit stream, which may be a subcomponent of the first component;
• In response to the format identification, selecting and invoking another one of the set of parsers to parse the second component.

15   This enables a selected parser of a set to control selection of other parsers to handle either subcomponents of the portion of a bit stream for which it has been given responsibility, or to invoke a next parser for a component of the bit stream subsequent to its assigned portion of the bit stream, both of which capabilities provide increased efficiency when
20   parsing a bit stream which includes multiple nested data formats.

The steps of a method according to an embodiment of the invention for parsing a data stream including multiple nested data formats can be summarised with reference to Figure 5. This shows an initial step of a
25   selector program component 80 identifying 200 the format of a first data component of the bit stream. The bit stream may include a field containing the class name of a specific parser for handling this data format or it may include data format information that can be compared with a repository holding information regarding which parsers are registered to handle which
30   formats. The information in the repository may be parser class names. The selector then instantiates 200 the named parser class to generate a specific instance of the appropriate parser to handle the current format. In alternative embodiments, the format of the first portion of all bit streams may be predefined for the particular system so that the appropriate
35   first parser can be invoked without format analysis. The selected parser begins parsing 210 the first component of the bit stream.

Two possible situations can now arise. If the bit stream has a single format, the selected parser parses the whole bit stream to output a model
40   230 of the data for subsequent processing. If the bit stream includes multiple formats, the first selected parser is responsible for identifying 240 the format of the component of the bit stream which follows the component it has parsed. This involves the parser analysing the contents of specific fields of the bit stream in a similar manner to the initial

GB920010021GB1   17

analysis performed by the selector component 80. The current parser then
calls a create parser method 240 for the appropriate parser class to invoke
a parser to handle this next component of the bit stream. This parser
parses its component 210, and the selection and parsing steps are repeated
5   for subsequent components having different formats. Each parser reports
what portion of the bit stream its parsing has handled so that the
information is available to determine where the next selected parser should
begin.

10   While this method is advantageous for a bit stream comprising
distinctly separable components, it is especially advantageous to enable
selected parsers to invoke a next parser when one component includes a
differently formatted subcomponent such that a parser handling the
component only requires a different parser to handle its subcomponent and
15   then requires control to be returned to it. In this case, a selected parser
is handed a chunk of the bit stream having a known size, it parses this
chunk and returns control to the parser which selected it. This provides
significant advantages compared with known alternatives when it is
necessary to parse a component in order to determine the format of (and
20   hence select a suitable parser for) its subcomponents.

CLAIMS

1.   A method of parsing a message containing a plurality of data formats,
the method comprising:

identifying the data format of a first component of the message;

responsive to said identification, selecting and invoking a first
parser to parse the first component; and

identifying the data format of a second component of the message
using said first selected parser and, responsive to said identification,
using said first selected parser to select and invoke a second parser for
parsing the second message component.

2.   A method according to claim 1, wherein identifying the data format of
the second component comprises reading a format field of the second
component, and wherein selecting and invoking a second parser comprises
comparing the identified format with a repository storing a list of parsers
associated with specific formats and storing format templates corresponding
to the specific formats, and providing the corresponding format template to
the second parser.

3.   A method according to claim 1 or claim 2, wherein invoking the second
parser includes inputting the second component to the second parser, and
the method further includes parsing the second component and then returning
control to the first selected parser.

4.   A method according to any one of claims 1 to 3, wherein the first
selected parser is adapted to parse a control structure of a message
component having an IDoc format, to read a format field of a data segment
of the IDoc component to identify the format, and to invoke a second parser
which is defined for parsing IDoc data segments having the identified
format.

5.   A method according to claim 4, wherein the first selected parser
passes the data segment to the second parser for parsing and then reads a
format field of a next component of the message to determine which parser
should parse the next component.

6.   A message processing system including a set of selectable parsers,
each selectable parser being adapted for analysing a respective set of
message data formats and being selectable in response to identifying a
message data format within the respective set, and a process for invoking a

GB920010021GB1   19

parser from the set, wherein at least one of said selectable parsers
includes:

   means for parsing a first component of a message having a message
5 data format within the respective set;

   means for identifying the data format of a second component of the
message; and

10   means, responsive to said identification, for selecting another one
of said set of parsers and for invoking the selected parser to parse the
second message component.

   7.   A message processing system according to claim 6, including a
15 repository of message format templates, each template representing the
field structure of a particular format of message component, wherein said
at least one parser is adapted to parse the first component to enable
identification of a format field containing information relating to the
format of the second component and to read the format information to
20 identify the data format, and wherein invoking the selected parser to parse
the second component includes providing to the selected parser an
identification of the message format template corresponding to the
identified format.

25   8.   A message processing system according to claim 6 or claim 7,
including a repository of message format templates, each template
representing the field structure of a particular format of message
component, wherein said at least one parser is adapted to parse the first
component and then to read a format field of a second component of the
30 message to identify the data format of the second component, and wherein
invoking the selected parser to parse the second component includes
providing to the selected parser an identification of the message format
template corresponding to the identified format.

35   9.   A data processing apparatus for processing a bit stream which may
contain a plurality of data formats, the apparatus including a set of
selectable parsers, each adapted for analysing a specific set of one or
more data formats, wherein a plurality of said set of parsers each include:

40   means for parsing a first component of a bit stream,

   means for identifying the data format of a second component of the
bit stream, and

GB920010021GB1   20

means, responsive to said identification, for selecting another one
of said set of parsers and for invoking the selected parser to parse the
second component.

5   10.   A data processing apparatus according to claim 9, wherein the
selected parser is adapted to access a format template from a format
dictionary corresponding to the format indication.

        11.   A data processing apparatus according to claim 9 or claim 10, wherein
10   the means for parsing the first component is adapted to output a name-value
pair indicating the format of the second component.

        12.   A data processing apparatus according to claim 9 or claim 10, wherein
the means for identifying the data format comprises means for analysing a
15   format field of the second component.

GB920010021GB1   21

ABSTRACT

PARSING MESSAGES WITH MULTIPLE DATA FORMATS

5   Provided are a method for parsing a bit stream including multiple
data formats, and an apparatus and computer program including a set of
parsers and parser-selection and invocation capabilities for handling
parsing of multiple data formats. A first parser is selected and invoked to
handle a first formatted component of the bit stream, and this selected
10   parser selects and invokes a next parser which is capable of handling a
differently formatted next component of the bit stream. This is
differentiated from systems which rely on a single generic parser or a
single high-level parser selection process, and is especially advantageous
when parsing messages to enable message processing in systems in which a
15   message can include multiple different, nested data formats.
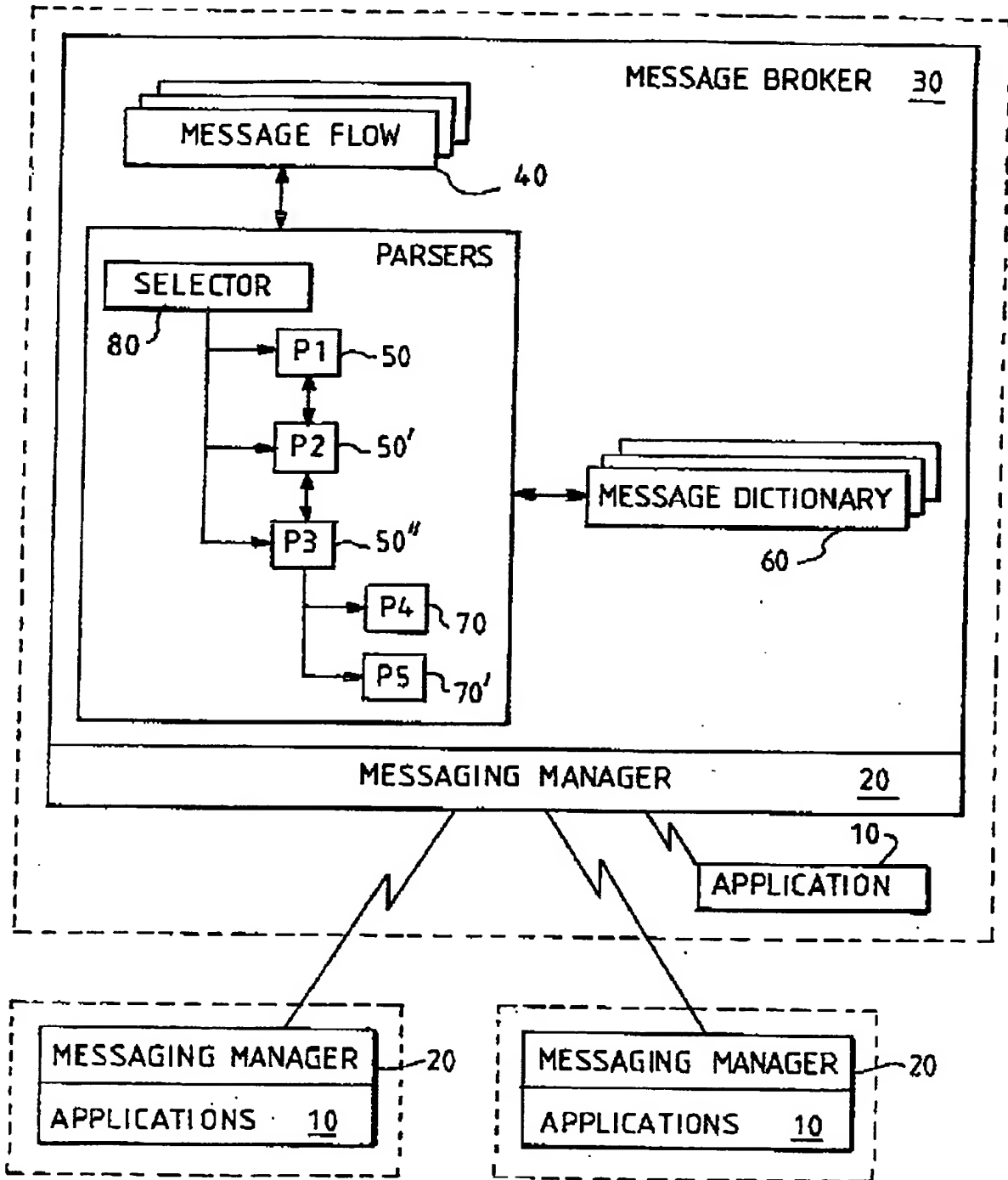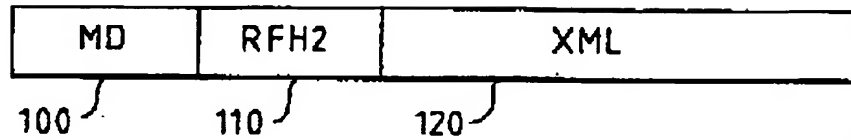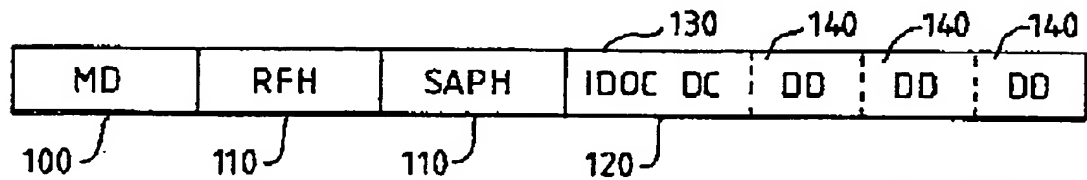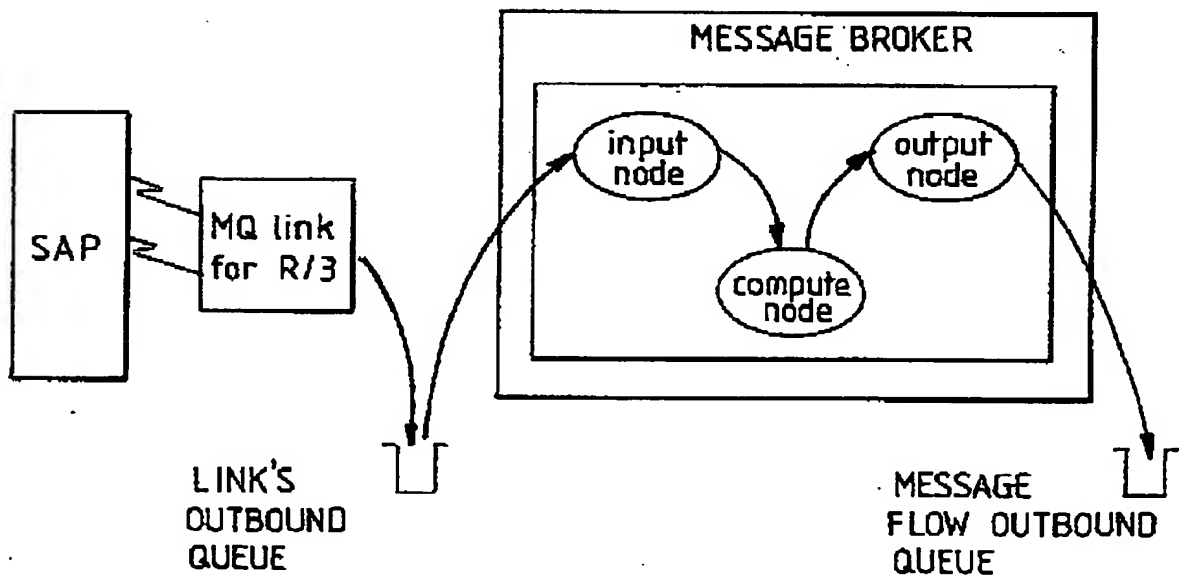
THIS PAGE BLANK (USPTO)

1/3
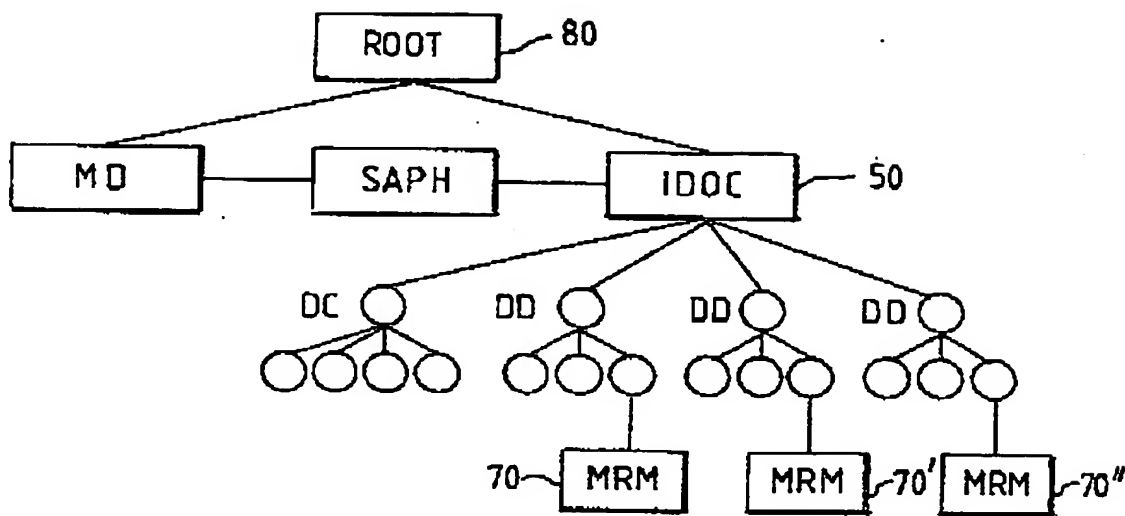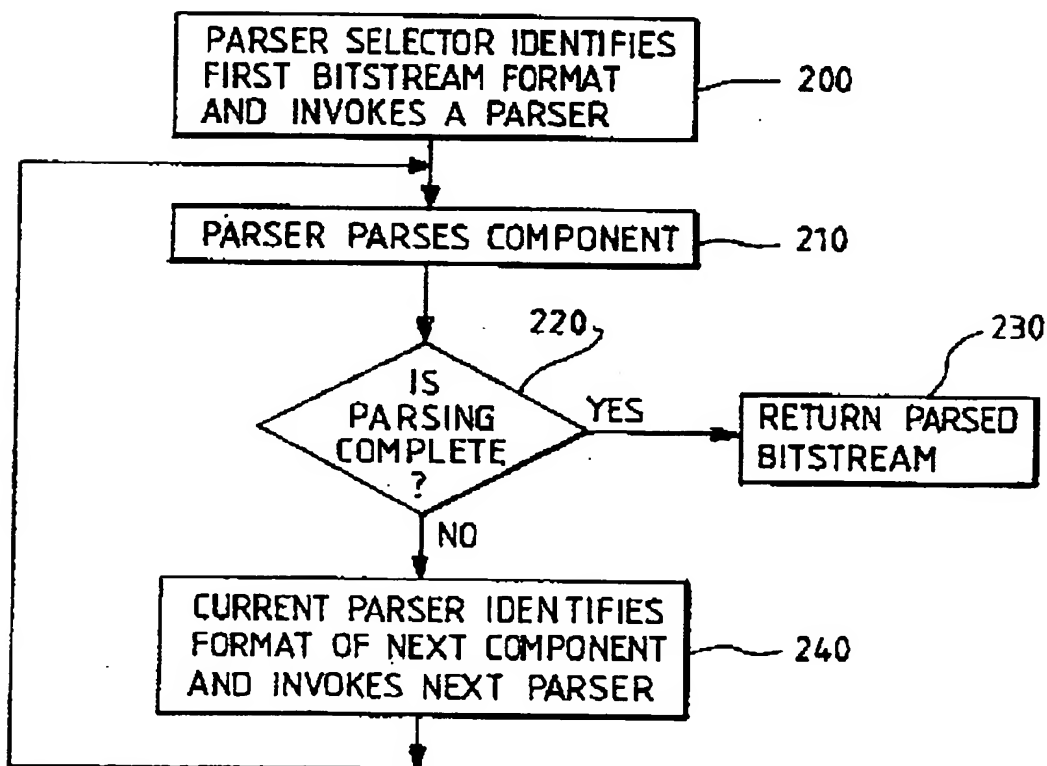

FIG. 1

THIS PAGE BLANK (USPTO)

2/3

MD	RFH2	XML

100   110   120

FIG. 2A

MD	RFH	SAPH	IDOC DC	DD	DD	DD

100   110   110   120   130   140 140 140

FIG. 2B

MESSAGE BROKER

input node

output node

compute node

SAP

MQ link for R/3

LINK'S OUTBOUND QUEUE

MESSAGE FLOW OUTBOUND QUEUE

FIG. 3

3/3

FIG. 4

FIG. 5